Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерный наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий.

**Отчёт по лабораторной работе №7**

**Дисциплина**: Проектирование мобильных приложений

**Тема**: Сервисы и Broadcast Receivers. Вариант 13

Выполнил студент гр. 3530901/80201: ­­­­­­­­­­­­­ ­ В.Д. Петров

(подпись)

Преподаватель: ­А.Н. Кузнецов

(подпись)

Санкт-Петербург

2020

**СОДЕРЖАНИЕ**

Введение…………………………………………………………………………….3

Задача №1. Started сервис для скачивания изображения…………………………5

Задача №2. Broadcast Receiver…………………………………………………....10

Задача №3. Bound Service для скачивания изображения………………………..12

Заключение………………………………………………………………………..19

Список использованных источников…………………………………………….20

**ВВЕДЕНИЕ**

Для процессов в Android операционная система выстраивает importance hierarchy: самыми важными считаются foreground процессы (процессы которые прямо сейчас нужны для того что делает пользователь), затем идут visible процессы (находятся на экране, но пользователь не может с ними взаимодействовать), затем идут сервисные процессы (не подпадают под предыдущие условия, однако имеют внутри себя запущенные сервисы), последними идут кэшированные процессы (они вообще не нужны и при необходимости освобождения оперативной памяти их можно будет спокойно удалить). Для сервисных процессов предусмотрено ограничение в 30 минут, после которого этот процесс может перейти в кэшированный и операционная система будет вправе его удалить из памяти. Activity нельзя оставлять background потоки, когда activity не отображается на экране, поскольку процесс этого activity в любой момент может перейти в состояние кэшированного и быть удаленным операционной системой. Как раз для таких ситуаций и были придуманы сервисы.

Сервис – это компонент приложения, который может выполнять длительные операции в фоновом режиме и не предоставляет пользовательского интерфейса. Сервис не является отдельным приложением или background потоком. Сервисы бывают двух типов – Started (разделяются на Foreground и Background) и Bound.

Started сервисы не возвращают никаких значений, ориентированы на выполнение некоторых задач, должны сами выполнить задачи и сами себя остановить, о Foreground процессах пользователь знает, например музыкальный плеер, который пользователь сворачивает, но хочет продолжать слушать музыку, Foreground процессы обязаны иметь нотификации, Background процессы выполняют задачи, которые пользователю не известны и не нужны, например очистка кэша.

Bound сервисы ориентированы на работу запрос-ответ, есть клиент, который выполняет какую то задачу на стороне сервиса и сервис возвращает результат в клиент. Bound сервисы будут остановлены операционной системой, когда все клиенты были обслужены.

У обоих сервисов немного похожий жизненный цикл (Рис. 1).

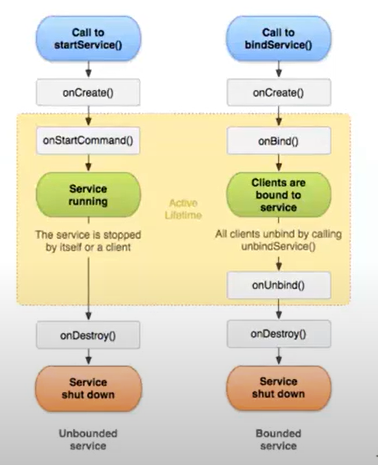


Рис. 1 Жизненный цикл сервисов

onCreate вызывается сразу после создания сервиса. onDestroy вызывается перед удалением сервиса. У Started сервисов есть метод onStartCommand, через который они получают информацию что нужно делать, у Bound сервисов два метода onBind и onUnbind через которые клиенты привязываются к сервисам и отвязываются от сервисов с помощью unbindService, после того как все клиенты отвяжутся от сервиса то будет вызван onUnbind.

Broadcast receiver – специальный компонент для получения broadcast сообщения (Intent, посылаемый ОС или любым приложением). С помощью этого компонента можно получать только те broadcast, которые соответствуют Intent фильтру этого компонента. Компонент выполняется только до завершения метода onReceive. Выполнение компонента происходит в UI потоке.

Целями данной работы является получение практических навыков разработки started и bound сервисов и broadcast receivers.

**ЗАДАЧА №1. STARTED СЕРВИС ДЛЯ СКАЧИВАНИЯ ИЗОБРАЖЕНИЯ**

Исходный код для этого задания находится по ссылке:

<https://github.com/ADsty/labs-android_labs/tree/master/lab7.1>

По заданию необходимо разработать Started сервис, который скачивает картинку из интернета. Для этого будем использовать Intent Service – он сам создает один background поток для обработки всех intent, после обработки последнего сам вызывает stopSelf и сам завершает запущенный background поток. Для использования этого класса необходимо переопределить метод onHandleIntent – в нем при получении нового намерения мы читаем, что записано в поле Extra (там будет передаваться ссылка на картинку), затем пытаемся скачать картинку и декодировать ее в bitmap – если попытка оказалась неудачной, то bitmap приравнивается к null. Затем вызывается метод saveImage, где мы сохраняем картинку во внутренней памяти и возвращаем путь к файлу с картинкой (при bitmap равным null возвращается null). В самом конце обработки намерения мы посылаем broadcast сообщение с помощью sendBroadcast, где будем передавать путь к файлу с картинкой.

Листинг 1 – исходный код для DownloadService

class DownloadService : IntentService("downloadService") {  
  
 override fun onHandleIntent(intent: Intent?) {  
  
 val url = intent?.getStringExtra("url")  
 var bitmap: Bitmap?  
 try {  
 bitmap = URL(url).openStream().*use* **{** return@use BitmapFactory.decodeStream(**it**)  
 **}** }catch (e: Exception){  
 bitmap = null  
 }  
  
 val path = saveImage(bitmap)  
  
 sendBroadcast(path)  
 }  
  
 private fun saveImage(bitmap: Bitmap?): String {  
 val nameOfImage = "myImage"  
 val stream: FileOutputStream  
 if(bitmap == null) return "null"  
 try {  
 stream = *applicationContext*.openFileOutput(nameOfImage, Context.*MODE\_PRIVATE*)  
 bitmap.compress(Bitmap.CompressFormat.PNG, 100, stream)  
 stream.close()  
  
 } catch (e: Exception) {  
 e.printStackTrace()  
 }  
  
 return *applicationContext*.getFileStreamPath(nameOfImage).*absolutePath* }  
  
 private fun sendBroadcast(msg: String) {  
 sendBroadcast(Intent("getPath").putExtra("path", msg))  
 }  
  
}

В main activity в методе onCreate создается Broadcast Receiver для того, чтобы получить путь к картинке и отобразить ее на экране, а также добавляется слушатель для кнопки – при нажатии ее будет создан intent с полем extra, где будет передана ссылка для картинки, после чего будет вызван сервис методом startService.

Листинг 2 – метод onCreate

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_start\_service*)  
  
 broadcastReceiver = object : BroadcastReceiver() {  
 override fun onReceive(context: Context?, intent: Intent?) {  
 val path = intent?.getStringExtra("path")!!  
 *println*(path)  
 val file = File(path)  
 if(file.exists()) {  
 image.setImageBitmap(BitmapFactory.decodeFile(path))  
 }  
 else{  
 Toast.makeText(context, "Can't Download", Toast.*LENGTH\_SHORT*).show()  
 }  
 load\_bar.*visibility* = View.*INVISIBLE* }  
 }  
  
 start\_service\_btn.setOnClickListener **{** startService(  
 Intent(this, DownloadService::class.*java*).putExtra("url", url)  
 )  
 load\_bar.*visibility* = View.*VISIBLE* **}**}

В методе onStart мы регистрируем broadcast receiver с нужным intent фильтром

Листинг 3 – метод onStart

override fun onStart() {  
 super.onStart()  
 registerReceiver(broadcastReceiver, IntentFilter("getPath"))  
}

В методе onStop мы отвязываем broadcast receiver

Листинг 4 – метод onStop

override fun onStop() {  
 super.onStop()  
 unregisterReceiver(broadcastReceiver)  
}

В результате получилось следующее приложение (Рис. 2).



Рис. 2 Результат запуска приложения

При нажатии кнопки будет запущен сервис и, если картинка будет успешно скачана – она отобразится на экране (Рис. 3).

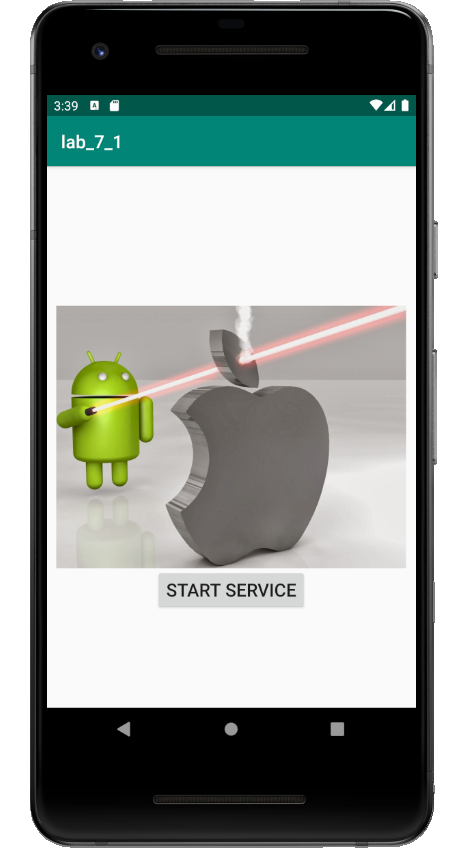


Рис. 3 Результат успешного скачивания картинки

Если картинку не удается скачать (например, неправильная ссылка), то ничего не отобразится на экране, а пользователю будет выдано сообщение, что не удалось скачать картинку (Рис. 4).

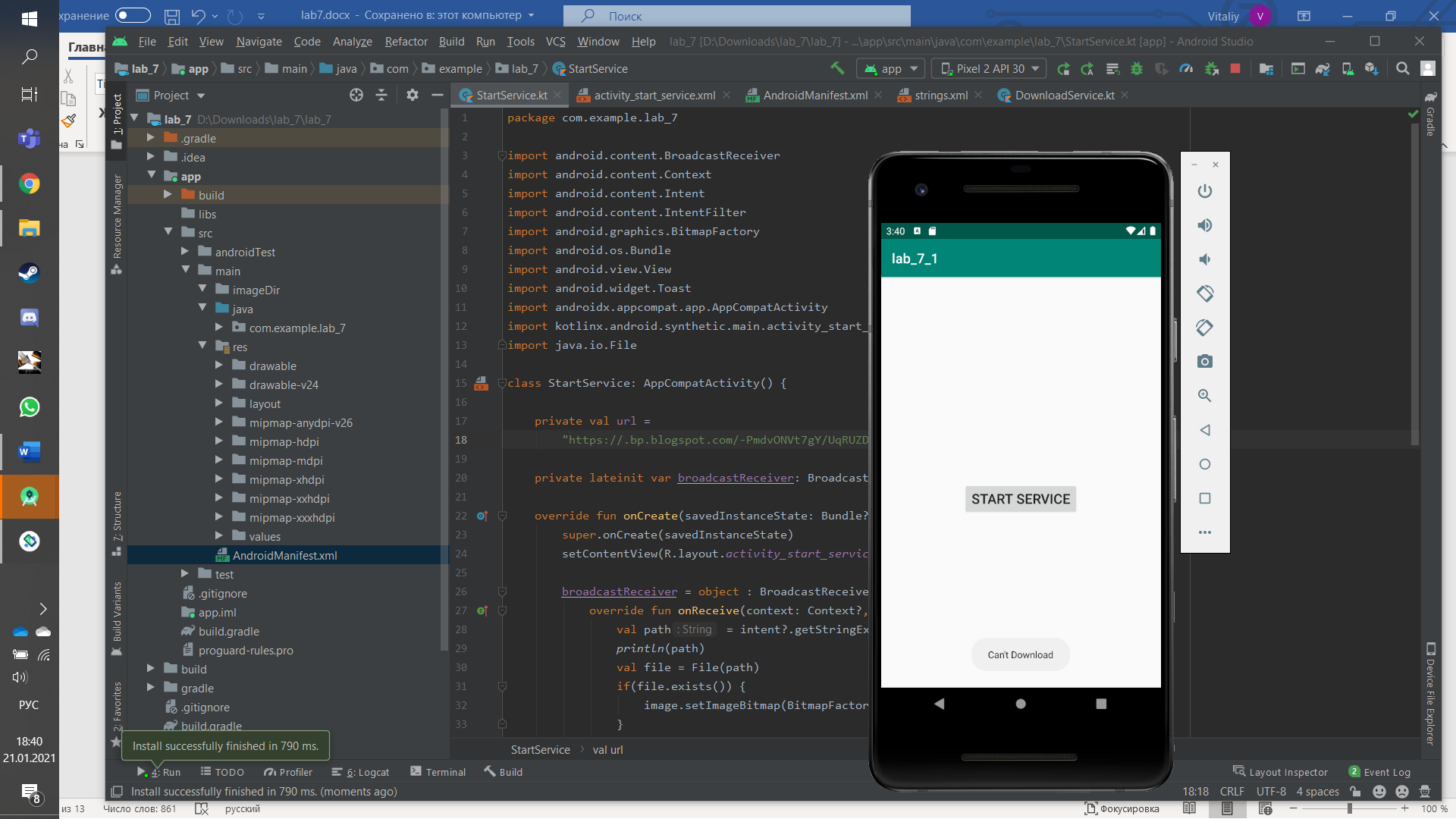


Рис. 4 Результат неудачного скачивания

Скачивания картинки в background потоке обеспечивается использованием Intent Service, он же завершает этот поток и вызывает внутри себя stopSelf.

**ЗАДАЧА №2. BROADCAST RECEIVER**

Исходный код для этой работы находится по ссылке:

<https://github.com/ADsty/labs-android_labs/tree/master/lab7.2>

Для данного задания необходимо разработать второе приложение, которое будет получать broadcast из предыдущего приложения и отображать путь к файлу с картинкой на экране. Для этого используется Broadcast Receiver, аналогичный такому же из предыдущего задания. Как и в прошлом задании мы регистрируем broadcast receiver с нужным intent фильтром и по необходимости отвязываем его. При первом запуске приложения будет выдано сообщение о необходимости запуска сервиса в первом приложении (Рис. 5).



Рис. 5 Первый запуск приложения

После сворачивания приложения и нажатия кнопки во втором приложении возвращаемся в наше приложение и на экране появляется путь к скачанной картинке (Рис. 6).



Рис. 6 Результат запуска сервиса в другом приложении

**ЗАДАЧА №3. BOUND SERVICE ДЛЯ СКАЧИВАНИЯ ИЗОБРАЖЕНИЯ**

Исходный код для этой работы находится по ссылке:

<https://github.com/ADsty/labs-android_labs/tree/master/lab7.3>

Для создания bound и started сервиса был использован Intent Service с уже написанным кодом для Started Service – теперь необходимо переопределить метод onBind, который должен возвращать объект класса IBinder. Данным объектом будет объект класса Messenger – в нем мы в качестве параметра передаем объект класса Handler с переопределенным методом handleMessage, в котором при получении сообщения от клиента будет вызван метод sendMessageToClient.

Листинг 5 – метод onBind

override fun onBind(intent: Intent): IBinder? {  
 fromClientMessenger = Messenger(  
 @SuppressLint("HandlerLeak")  
 object : Handler() {  
 override fun handleMessage(message: Message) {  
 when (message.what) {  
  
 messageFromClient -> {  
 sendMessageToClient(message)  
 }  
 }  
 }  
 })  
 return fromClientMessenger.*binder*}

В методе sendMessageToClient мы получаем URL картинки, пытаемся ее скачать, используя корутины (из main потока вызываем async корутину с Dispatchers.IO для работы в интернете). Таким образом мы гарантируем, что скачивание картинки происходит не в UI потоке. После этого мы вызываем метод saveImage, который сохраняет картинку в файле и возвращает путь к ней, этот путь мы отправляем в сообщении, которое получит Messenger на стороне activity.

Листинг 6 – метод sendMessageToClient

private fun sendMessageToClient(messageFromClient: Message): Bitmap? {  
 val url = messageFromClient.*data*.getString("url", url)  
 var bitmap: Bitmap? = null  
 val toClientService = messageFromClient.replyTo  
  
 *CoroutineScope*(Dispatchers.Main).*launch* **{** bitmap = *async*(Dispatchers.IO) **{** try {  
 URL(url).openStream().*use* **{** return@async BitmapFactory.decodeStream(**it**)  
 **}** }catch (e: Exception){  
 return@async null  
 }  
 **}**.await()  
  
 val path = saveImage(bitmap)  
  
 val message = Message.obtain(null, toClient, 0, 0).*apply* **{** *data* = Bundle().*apply* **{** putString("answer", path)  
 **}  
 }** toClientService.send(message)  
 **}** return bitmap  
}

Метод saveImage идентичен методу из первого задания. Из первого задания так же перешли методы onHandleIntent и sendBroadcast для работы Started версии сервиса. В main activity была добавлена ссылка на картинку из лабораторной работы № 6 – ее мы будем скачивать с помощью Bound сервиса, а Started сервис будет скачивать предыдущую картинку. Также был добавлен свой Messenger для обработки сообщений из Bound сервиса – при получении сообщения приложение попытается найти файл со скачанной картинкой и в случае успеха поставит картинку заместо старой (пустой или же скачанной ранее Bound или Started сервисом)

Листинг 7 – Messenger для main activity

private val fromServiceMessenger = Messenger(CaughtMessages())  
  
@SuppressLint("HandlerLeak")  
inner class CaughtMessages : Handler() {  
  
 override fun handleMessage(message: Message) {  
 when (message.what) {  
 messageFromService -> {  
 val path = message.*data*.getString("answer")!!  
 val file = File(path)  
 if(file.exists()) {  
 image.setImageBitmap(BitmapFactory.decodeFile(path))  
 }  
 else{  
 Toast.makeText(this@MainActivity, "Can't Download", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
 }  
}

Была добавлена кнопка для Bound сервиса – в ней мы отправляем сообщение для Messenger в сервисе с прикрепленной ссылкой на картинку и указываем получателя для писем с помощью поля replyTo, где указываем fromServiceMessenger

Листинг 8 – исходный код для второй кнопки

lab7\_3\_button.setOnClickListener **{** val message = Message.obtain(null, messageFromClient, 0, 0)  
  
 message.*data* = Bundle().*apply* **{** putString("url", url2)  
 **}** message.replyTo = fromServiceMessenger  
  
 toServiceMessenger?.send(message)  
**}**

В методе onStart мы привязываемся к Bound сервису с помощью метода bindService.

Листинг 9 – метод onStart

override fun onStart() {  
 super.onStart()  
  
 registerReceiver(broadcastReceiver, IntentFilter("getPath"))  
  
 val intent = Intent(this, DownloadService::class.*java*)  
 this.bindService(intent, connectionOfService, Context.*BIND\_AUTO\_CREATE*)  
}

В методе onStop мы отвязываемся от сервиса

Листинг 10 – метод onStop

override fun onStop() {  
 super.onStop()  
  
 unregisterReceiver(broadcastReceiver)  
  
 this.unbindService(connectionOfService)  
}

Для взаимодействия с Bound сервисом был создан объект ServiceConnection – в методе onServiceConnected мы получаем ссылку на IBinder для взаимодействия с сервисом, в методе onServiceDisconnected мы приравниваем ссылку к null.

Листинг 11 – объект ServiceConnection

private val connectionOfService = object : ServiceConnection {  
  
 override fun onServiceDisconnected(name: ComponentName) {  
 toServiceMessenger = null  
 }  
  
 override fun onServiceConnected(name: ComponentName, service: IBinder) {  
 toServiceMessenger = Messenger(service)  
 }  
}

В результате запуска появляется приложение с двумя кнопками (Рис. 7).



Рис. 7 Старт приложения

При нажатии на первую кнопку скачивается картинка из 1 задачи с помощью Started сервиса (Рис. 8).

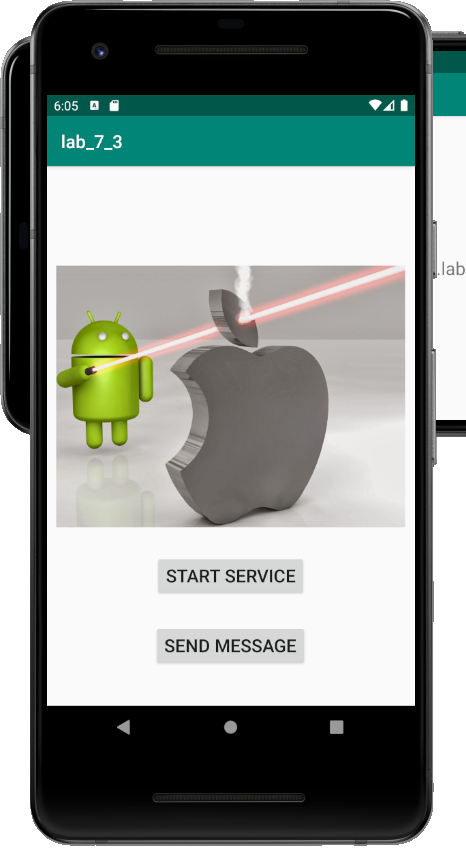


Рис. 8 Результат нажатия на первую кнопку

При нажатии второй кнопки скачивается и появляется на экране картинка из 6 лабораторной работы (Рис. 9).

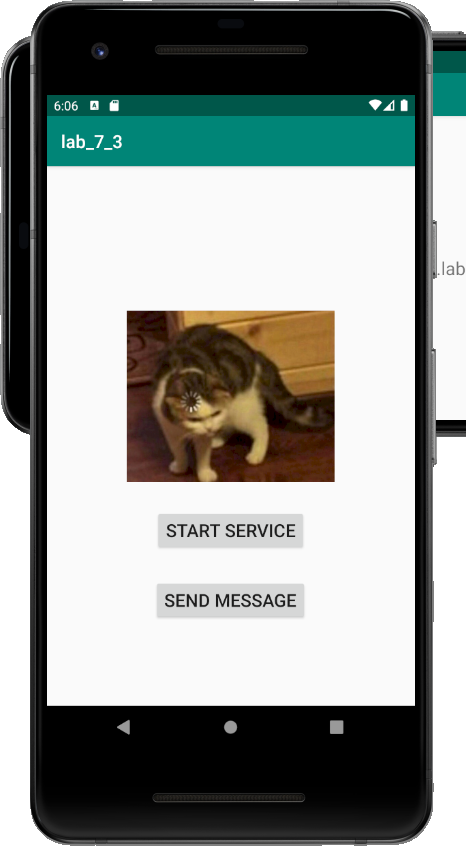


Рис. 9 Результат нажатия второй кнопки

При неудачной попытке скачивания картинки в обоих случаях на экране картинка не поменяется, а пользователь будет уведомлен, что скачивание не удалось (Рис. 10).

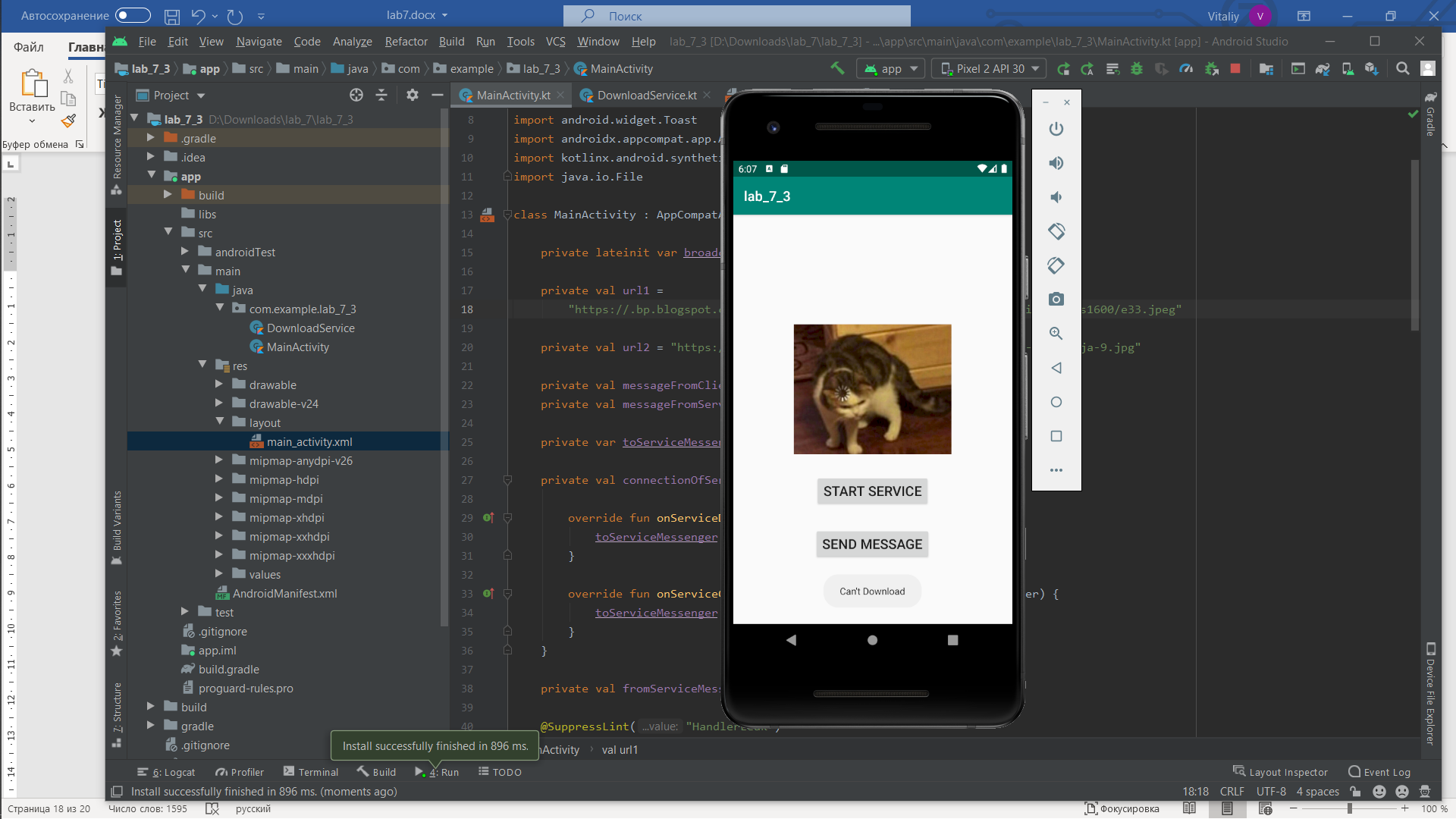


Рис. 10 Результат неудачного скачивания картинки

**ЗАКЛЮЧЕНИЕ**

В данной лабораторной работе были получены практические навыки разработки разных видов сервисов: сначала был разработан Started сервис на основе Intent Service, затем было разработано приложение, получающее из этого сервиса информацию о пути к скачанной картинке с помощью broadcast receiver, и, наконец, первое приложение было дополнено и на основе Intent Service был сделан bound и started сервис. В нем был переопределен метод onBind, внутри которого использовался Messenger. Ситуации вроде неудачного скачивания картинки также были обработаны – в этом случае пользователь получает уведомление о неудаче и на экране картинка не обновляется.

Исходный код для лабораторных работ находится на [github](https://github.com/ADsty/labs-android_labs) (<https://github.com/ADsty/labs-android_labs> )

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

<https://developer.android.com/guide/components/services?hl=ru>

<https://developer.android.com/guide/components/broadcasts?hl=ru>

<https://developer.android.com/training/data-storage/app-specific?hl=ru>